

UNITED STATES PATENT APPLICATION FOR

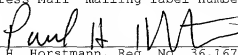
MULTI-LAYER SOFTWARE ARCHITECTURE
FOR HARDWARE CONTROL

Inventors:
Brian Frazier
Keith Sutton
Thanh Heyman

CERTIFICATE OF MAILING BY "EXPRESS MAIL"
UNDER 37 C.F.R. § 1.10

"Express Mail" mailing label number: EI5609874575US
Date of Mailing: 4-4-2001

I hereby certify that this correspondence is being deposited with the United States Postal Service, utilizing the "Express Mail Post Office to Addressee" service addressed to **Assistant Commissioner for Patents, Washington, D.C. 20231** and mailed on the above Date of Mailing with the above "Express Mail" mailing label number.


Paul H. Horstmann, Reg. No. 36,167
Signature Date: 4-4-2001

BACKGROUND OF THE INVENTION

Field of Invention

5 The present invention pertains to the field of software control of hardware. More particularly, this invention relates to a multi-layer software architecture for hardware control.

Art Background

10 A wide variety of systems include hardware which is controlled by software. Software that controls hardware is commonly referred to as firmware. For example, a wide variety of instruments including those used in test and measurement applications
15 commonly include circuits for generating test signals and/or circuits for obtaining measurements along with firmware for controlling the circuits.

20 A circuit in such a system typically includes one or more control points that enable software to control a circuit by writing values to the control points of the circuit. The control points may be implemented as registers and/or digital-to-analog converters, etc.

25 It is common to make hardware changes to a system when performing support or upgrade operations on the system. Hardware changes to a system may involve modification to its circuits, replacement of
30 its circuits, and/or the addition of circuits. Such hardware changes typically require modifications to the firmware that controls the circuits.

Prior firmware systems may grow quite large and complex as the number and complexity of circuits being controlled increases. In addition, the firmware code that directly accesses the control points of circuits is usually scattered haphazardly throughout numerous portions of a firmware system. These factors typically complicate the task of performing firmware modifications. For example, it is common to undertake a search through large amounts of code to find references to the control points that require modification. Unfortunately, such methods of performing firmware modification are usually time consuming and can greatly increase the time and cost of upgrading a system and may introduce faults into the firmware.

SUMMARY OF THE INVENTION

5 A software system having a multi-layer
architecture for controlling a hardware system is
disclosed. The multi-layer architecture includes a
latch layer, a hardware control layer, an access
layer, and an orchestration layer. The latch layer
includes a latch object for each of a set of control
10 points of the hardware system. Each latch object
provides a common interface in the software system
for accessing the corresponding control point. The
hardware control layer includes a hardware control
object for each of a set of sub-portions of the
hardware system. Each hardware control object
15 coordinates accesses to the control points of the
corresponding sub-portion through the latch layer.
The access layer includes an access object for each
of a set of groupings of the sub-portions. Each
access object coordinates accesses to the
20 corresponding grouping of the sub-portions. The
orchestration layer includes an orchestration object
for each of a set of functional features of the
hardware system. Each orchestration object provides
a common interface in the software system for
25 accessing a corresponding grouping of the access
objects which are associated with the corresponding
functional feature.

30 Other features and advantages of the present
invention will be apparent from the detailed
description that follows.

BRIEF DESCRIPTION OF THE DRAWINGS

5 The present invention is described with respect
to particular exemplary embodiments thereof and
reference is accordingly made to the drawings in
which:

10 **Figure 1** shows one embodiment of a software
system according to the present teachings which
includes a set of latch objects grouped by a hardware
control object;

15 **Figure 2** shows another embodiment of a software
system according to the present teachings which
includes an access object for grouping hardware
control objects;

20 **Figure 3** shows yet another embodiment of a
software system according to the present teachings
which includes an orchestration object for grouping
access objects;

Figure 4 shows still another embodiment of a
software system according to the present teachings.

DETAILED DESCRIPTION

5 **Figure 1** shows one embodiment of a software system 100 according to the present teachings. The software system 100 controls a hardware subsystem 200-1 which may be a sub-portion of a hardware system. The hardware subsystem 200-1 includes a set of control points 50-1 through 50-n for controlling the functionality of the hardware subsystem 200-1.

10 The software system 100 includes a set of latch objects 40-1 through 40-n which share a common interface and form a latch layer 500. The latch objects 40-1 through 40-n may share a similar responsibility with respect to the control points 50-1 through 50-n.

15 The latch objects 40-1 through 40-n correspond to the control points 50-1 through 50-n, respectively. The latch objects 40-1 through 40-n encapsulate and provide common interfaces to the corresponding control points 50-1 through 50-n, respectively. For example, the latch object 40-1 encapsulates and provides a common interface to the control point 50-1 and the latch object 40-2 encapsulates and provides a common interface to the control point 50-2, etc.

20 Each latch object 40-1 through 40-n provides a wrapper and locking mechanism around a physical address associated with the corresponding control point 50-1 through 50-n. For example, the latch object 40-1 provides a wrapper and locking mechanism

around the physical address associated with the control point 50-1 and the latch object 40-2 provides a wrapper and locking mechanism around the physical address associated with the control point 50-2, etc.

5

In one embodiment, each latch object 40-1 through 40-n is an object implemented as a C++ class and its common interface includes a set of methods. The methods include a change_state() method for altering the corresponding control point 50-1 through 50-n. For example, the change_state() method of the latch object 40-1 takes as an argument a value to be applied to the control point 50-1 and the change_state() method of the latch object 40-2 takes as an argument a value to be applied to the control point 50-2. Each latch object 40-1 through 40-n uses a C++ locking mechanism to lock the physical address associated with the corresponding control points 50-1 through 50-n.

20

The implementation of the latch objects 40-1 through 40-n are adapted to the particular hardware implementation of the underlying control points 50-1 through 50-n and insulate higher objects of the software system 100 from these particulars. For example, an alteration of the circuitry associated with the control point 50-2 can be accommodated by a modification to the latch object 40-2 rather than modifications to other portions of the software system 100.

30

The software system 100 includes a hardware control object 30-1 that encapsulates and provides a

common interface for controlling the hardware
subsystem 200-1. The hardware control object 30-1
provides a hardware control layer 510 in the software
system 100. The hardware control object 30-1
5 coordinates the latch objects 40-1 through 40-n.

The hardware control object 30-1 includes a set
of methods which are adapted to provide a set of
hardware functions associated with the hardware
10 subsystem 200-1 according to the underlying hardware
implementation. The methods in the hardware control
object 30-1 are adapted to calculate values to be
applied to the control points 50-1 through 50-n and
call the change_state() methods of the latch objects
15 40-1 through 40-n when appropriate. Code in higher
objects of the software system 100 call the methods
of the hardware control object 30-1 when accessing
the hardware functions provided by the hardware
subsystem 200-1. In one embodiment, the hardware
20 control object 30-1 is an object implemented as a C++
container class for the latch objects 40-1 through
40-n.

In some embodiments, the latch objects 40-1
25 through 40-n are grouped together using the hardware
control object 30-1 because the control points 50-1
through 50-n may have interdependencies with respect
to one another. The nature of the interdependencies
among the control points 50-1 through 50-n may depend
30 on the particular application-specific function of
the hardware subsystem 200-1. One example of an
interdependency between a pair of control points is
one in which the valid range of values for one

depends on the value of the other. The coding of the methods in the hardware control object 30-1 are adapted to the interdependencies and shield higher levels of the software system 100 from having to adapt to the interdependencies. An alteration or replacement of the hardware subsystem 200-1 including changes in the interdependencies among the control points 50-1 through 50-n can be accommodated by a modification to the hardware control object 30-1 rather than modifications to other portions of the software system 100.

In some embodiments, the hardware control object 30-1 may include files having calibrated data to be applied to the control points 50-1 through 50-n.

Figure 2 shows another embodiment of the software system 100 according to the present teachings. The software system 100 in this embodiment controls the hardware subsystem 200-1 along with a hardware subsystem 200-2 each of which is a sub-portion of a hardware system 300. The hardware subsystem 200-2 includes a set of control points 150-1 through 150-n for controlling its functionality.

As before, the software system 100 includes the hardware control object 30-1 for grouping together the latch objects 40-1 through 40-n which are associated with the control points 50-1 through 50-n of the hardware subsystem 200-1. In this embodiment, the hardware control layer 510 of the software system 100 includes a hardware control object 30-2 for

grouping together a set of latch objects 140-1 through 140-n which are associated with the control points 150-1 through 150-n of the hardware subsystem 200-2.

5

The latch objects 140-1 through 140-n encapsulate accesses to the control points 150-1 through 150-n, respectively, in a manner substantially similar to that described with respect to the latch objects 40-1 through 40-n. The hardware control object 30-2 encapsulates and provides a common interface to the functionality of the hardware subsystem 200-2 in a manner substantially similar to that described with respect to the hardware control object 30-1.

15

The software system 100 in this embodiment includes an access layer 520 having an access object 20-1 that encapsulates and coordinates accesses to the hardware subsystems 200-1 and 200-2 in response to requests for specific hardware actions from higher levels of the software system 100. The access object 20-1 implements methods that coordinate the hardware functions associated with the hardware control objects 30-1 and 30-2. In this context, the access object 20-1 groups together the hardware subsystems 200-1 and 200-2. The grouping may be based upon interdependencies among hardware subsystems 200-1 and 200-2.

20

25

30

Consider an example in which the subsystem 200-1 controls the frequency of a waveform and the subsystem 200-2 attenuates the waveform and in which

there is a dependency between the frequency of the waveform and the amount of attenuation. Code in the access object 20-1 may call the methods of the hardware control object 30-2 to obtain an attenuation setting from the subsystem 200-2, determine an appropriate frequency setting according to an interdependency, and then call the methods of the hardware control object 30-1 to set the frequency of the waveform accordingly.

Figure 3 shows yet another embodiment of the software system 100 according to the present teachings. The software system 100 in this embodiment controls the hardware subsystems 200-1 and 200-2 along with a hardware subsystem 1200-1. The hardware subsystems 200-1 and 200-2 and the hardware subsystem 1200-1 are each a sub-portion of the hardware system 300.

As before, the software system 100 includes the hardware control objects 30-1 and 30-2 for providing a common interface to the latch objects 40-1 through 40-n and the latch objects 140-1 through 140-n, respectively. Similarly, the software system 100 includes the access object 20-1 for grouping together the hardware control objects 30-1 and 30-2. In this context, the access object 20-1 groups together and provides a common interface to the functionality of the hardware subsystems 200-1 and 200-2.

In this embodiment, the software system 100 includes a hardware control object 130-1 for grouping together a set of latch objects 240-1 through 240-n

which are associated with a set of corresponding control points of the hardware subsystem 1200-1. The software system 100 in this embodiment includes an access object 20-2 that encapsulates, provides a common interface to, and coordinates accesses to the hardware subsystem 1200-1.

The hardware control objects 30-1, 30-2 and 130-1 of the hardware control layer 510 may be viewed as sharing a similar responsibility in that they coordinate latch objects. The access objects 20-1 and 20-2 of the access layer 520 may be viewed as sharing a similar responsibility in that they coordinate hardware control objects.

The software system 100 in this embodiment includes an orchestration layer 530 having an orchestration object 10-1 for grouping together and providing a common interface to the access objects 20-1 and 20-2. The grouping of the access objects 20-1 and 20-2 may be based on interdependencies in the functionality of their underlying hardware subsystems. The grouping of the access objects 20-1 and 20-2 may define a functional feature of the hardware system 300. The orchestration object 10-1 may implement methods that provide higher level feature-based functionality which is substantially independent of the underlying hardware system 300. In one embodiment, the orchestration object 10-1 is an object implemented as a C++ class having data that includes pointers to the access objects 20-1 and 20-2.

Consider an example in which the hardware system 300 generates a waveform, the hardware subsystem 200-1 controls the frequency of the waveform and the hardware subsystem 200-2 attenuates the waveform and in which the hardware subsystem 1200-1 conditions the waveform. In this context, the access object 20-1 encapsulates the functions in the software system 100 that are involved in generating the source waveform and the access object 20-2 encapsulates the functions that are involved in conditioning the waveform. The orchestration object 10-1 calls methods in the access object 20-1 to control the generation of the source waveform and calls methods in the access object 20-2 to control the conditioning of the source waveform.

The hardware system 300 and its sub-portions - the hardware subsystems 200-1, 200-2, and 1200-1 may be embodied in a variety of arrangements. For example, the hardware system 300 may be a circuit board or module and the hardware subsystems 200-1 and 200-2 and 1200-1 may be separate circuits contained on it. In another example, the hardware subsystems 200-1 and 200-2 and 1200-1 may be separate sub-circuits of a larger circuit that is the hardware system 300. In yet another example, the hardware subsystems 200-1 and 200-2 and 1200-1 may be separate modules or circuit boards contained within a rack mounted hardware system 300. The control points of the hardware subsystems 200-1, 200-3, and 1200-1 may be implemented as registers or digital-to-analog converters, etc., or any combination of these.

The hierarchical arrangement of the hardware system 300 and the hardware subsystems 200-1, 200-2, and 1200-1 and the control points in the hardware subsystems 200-1, 200-2, and 1200-1 correspond to the hierarchical arrangement of the access objects 20-1 and 20-2, the hardware control objects 30-1, 30-2, and 130-1, and the latch objects 40-1 through 40-n, 140-1 through 140-n, and 240-n through 240-n. Changes that encompass a grouping of hardware subsystems are handled by changes to a corresponding access object and below. Changes that encompass a hardware subsystem are handled by changes to a corresponding hardware control object and below. Changes that encompass a control point are handled by changes to a corresponding latch object.

For example, the software system 100 adapts to a modification or replacement of the hardware subsystems 200-1 and 200-2 with a modification or replacement of the access object 20-1. The software system 100 adapts to a modification or replacement of the hardware subsystem 200-1 with a modification or replacement of the hardware control object 30-1. The software system 100 adapts to a modification or replacement of the control point 50-1 with a modification or replacement of the latch object 40-1. The modification to a object may include modifications to its data and/or its methods.

Figure 4 shows still another embodiment of the software system 100 according to the present teachings. The software system 100 in this embodiment includes the orchestration object 10-1,

the access objects 20-1 and 20-2, and the hardware control objects 30-1, 30-2, and 130-1 along with corresponding latch objects for controlling the hardware subsystems 200-1, 200-2 and 1200-1.

5

In this embodiment, the software system 100 includes a hardware control object 1030-1 and an access object 120-1 for a corresponding hardware subsystem and a hardware control object 1030-2 and an
10 access object 120-2 for a corresponding hardware subsystem and an orchestration object 10-2 for grouping the access objects 120-1 and 120-2.

The orchestration object 10-1 handles requests
15 from a user which are associated with a type A function feature of the hardware system 300 and the orchestration object 10-2 handles requests from the user which are associated with a type B function feature of the hardware system 300. Consider an
20 example in which the orchestration object 10-1 uses the access objects 20-1 and 20-2 to control the frequency of a waveform and the orchestration object 10-2 uses the access objects 120-1 and 120-2 to control the amplitude of the waveform. In this
25 context, type A functional features are frequency related and type B functional features are amplitude related. The orchestration object 10-1 may include a method that takes as an argument a frequency value and the orchestration object 10-2 may have a method
30 that takes as an argument an amplitude value.

Each latch object is controlled by only one hardware control object and each hardware control

object is controlled by only one access object whereas multiple orchestration objects can control each access object. In addition, orchestration objects can control other orchestration objects.

5 This enables an access object to handle conflicts in the control undertaken by their orchestration objects. For example, the access objects 20-2 and 120-1 are each contained in both the orchestration objects 10-1 and 10-2. If the orchestration object 10-1 undertakes to generate a frequency of 500 MHZ and the orchestration object 10-2 undertakes to set an amplitude of 10V then either the access object 20-2 or the access object 120-1 may determine that the corresponding hardware is only capable of 5V and 500
10 10-1 undertakes to generate a frequency of 500 MHZ and the orchestration object 10-2 undertakes to set an amplitude of 10V then either the access object 20-2 or the access object 120-1 may determine that the corresponding hardware is only capable of 5V and 500
15 MHZ and take appropriate action to apply these constraints.

No communication is allowed between access objects. This forces all requests to be coordinated by an orchestration object. Similarly, no
20 communication is allowed between hardware control objects.

In some embodiments, communication is allowed
25 between orchestration objects. For example, the orchestration object 10-1 communicates to the orchestration object 10-2 that it is changing the frequency of the waveform which is an operation that may require the orchestration object 10-2 to change
30 an operating mode if its underlying hardware.

In one embodiment, the software system 100 includes initialization or boot-up code for detecting

the versions of the hardware systems and subsystems installed and for setting the pointers maintained by each orchestration objects to its corresponding access objects and for setting the pointers maintained by each access objects to its corresponding hardware control objects accordingly.

The multi-layer architecture disclosed herein may be used to impose rules on the laying out of control of hardware systems including embedded systems. These rules provide clarity and consistency in generating new control algorithms for a system as well as maintenance of a system.

The orchestration layer enables the implementation of high-level, feature-based algorithms in which there is little need for knowledge of the underlying hardware system. The access layer provides for encapsulation of hardware function circuits and enables implementation of control which is specific to the underlying circuitry. The access layer provides an application programming interface (API) to the underlying circuitry and enables use of the underlying circuit functionality without specific knowledge of the design of the underlying circuitry. The hardware control layer is provided to abstract the specific digital interface to the underlying circuitry from the access layer. Minor changes to a portion of underlying circuitry may be changed while not affecting the access layer. For example, the resolution of a control DAC may be changed with a corresponding change to the code in the hardware

control layer while not affecting the code in the
access layer

The foregoing detailed description of the present invention is provided for the purposes of illustration and is not intended to be exhaustive or to limit the invention to the precise embodiment disclosed. Accordingly, the scope of the present invention is defined by the appended claims.